LAB #10
Wrapping Up the Fall Quarter

Check your grades on BB to make sure they are accurate!!! We do not have more labs, and you need to make sure your grade is correct!!!

Make sure you have Assignments #1 - #8 graded because this is the last week the TAs will hold office hours for grading or helping anyone!!! We will grade Assignment #9 during finals week. You do not need to demo Assignment #9!!!

In this course, all our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab, as specified in the student handout. At this time, you need to pair with someone in the lab, and finish the rest of the lab as a pair.

(5 pts) Code the design/pseudocode

# Description: This is where execution of the program begins. The main function will get # the number of verses to sing from the user, and then sing the song with that many # verses.
# Parameters: none
# Preconditions: none
# Postconditions: none
# Return: none
main()
    Prompt the user for the number of verses
    Read the number of verses as a string
    if the string is an integer, send string to is_int()
    send the converted integer to sing_song()
    else
        print an error message

# Description: This function checks to see if a string is an int
# Parameters: s
# Preconditions: s is a string
# Postconditions: none
# Return: bool, true if it is an int and false otherwise
is_int(s)
    for all the characters in s
        if (the character is not between ‘0’ and ‘9’ inclusive) and (it is not the first or (it is the first character and not equal to ‘-’))
            return false

    return true
# Description: This function sings the song, “99 bottles of beer on the wall”, but it starts
# at a specific verse/number.
# Parameters: n
# Preconditions: n is an integer value
# Postconditions: none
# Return: none

```python
def sing_song(n):
    while n verses is greater than 0:
        print "n bottles of beer on the wall. Take one down, pass it around. n-1 
bottles of beer on the wall."
        decrement n by 1
```

(5 pts) Code the design/flowchart

```
main()
  get string of parenthesis
  if string is balanced then
    print yes
    is_balanced(s)
    print no
  else
    print no message
```

Description - this is where execution begins and it gets a string from the user and checks if the string is balanced parenthesis.
Description - checks to see if string is balanced parenthesis. For example:
( ), ((())), ()() are balanced
(, ), )(, )((())) , )() are not balanced
Parameters: s
Preconditions: none
Postconditions: none
Return: True if string is balanced parenthesis
False otherwise

is-balanced(s)

set open-paren to 0

if we are not at the end of the string

if the character is ')' and open-paren is zero

decrement open-paren by one

increment open-paren by one

if open-paren to 0

if open-paren to

return False

return True

return False

else if character is '(',

else if character is '(',

return False